VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF APPLIED SCIENCE

PROBABILITY AND STATISTICS (MT2013)

Probability and Statistics Assignment - Semester 232

The impact of GPU's characteristics on its Memory Speed

Class CC07 - Group 09

Advisor: Dr. Phan Thi Huong
Students: Huynh Nguyen Ngoc Anh - 2252022
Nguyen Minh Khoi - 2252377
Dinh Ba Khanh - 2252323
Nguyen Hoang Vu - 2252919
Tran Thi Hong Hanh - 2252193

HO CHI MINH CITY, MAY 2024

Member list & Workload

No.	Fullname	Student ID	Task	Contribution
1	Huynh Nguyen Ngoc Anh	2252022	Write report, research and explain mathematical model for coding	100%
2	Nguyen Minh Khoi	2252377	Coding Sections: Data preparation, support in Linear Regression	100%
3	Dinh Ba Khanh	2252323	Coding Sections: Data preproceeding, Linear Regression, Random Forest Regression	100%
4	Nguyen Hoang Vu	2252919	Coding Sections: Data Descriptive, support in Random Forest regression	100%
5	Tran Thi Hong Hanh	2252193	Search for topic, write report, research and explain mathematical model for coding	100%

Leader contact information: Dinh Ba Khanh - khanh.dinhcse@hcmut.edu.vn

Comments and Evaluation

Comments	Evaluation

Contents

1 Data Introduction

1.1 Dataset Overview

The graphics processing unit, or GPU, has become one of the most important types of computing technology, both for personal and business computing. Designed for parallel processing, the GPU is used in a wide range of applications, including graphics and video rendering. Although they're best known for their capabilities in gaming, GPUs are becoming more popular for use in creative production and artificial intelligence (AI).

This project aims at analyzing some of GPU's characteristics perform in the given dataset. The table below shows a summary of the GPU dataset:

Properties	**Information**
Data Source	All_GPUs.csv
Population	Collection of GPUs
Number of observation	3406
Number of variables	34

Table 1: *Overview of dataset*

1.2 Variables Overview

The table below shows the overview of variables in the dataset.

Variable	**Data type**	**Unit**	**Description**
Architecture	categorical		Codename classification for a GPU microarchitecture
Best Resolution	categorical		The highest level of detail in an image which is determined by the number of pixels it contains.
Boost Clock	continuous	MHz	Temporary high speed at which GPU operates when it is under heavy load
Core Speed	continuous	MHz	The speed which the card's main processor operates
DVI Connection	categorical		The number of video connector used to transmit digital video signals from a computer or other device to a display screen
Dedicated	categorical		Graphics card which has separate component from CPU with its own memory and processing unit
Direct X	categorical		A collection of application programming interfaces (APIs) for handling tasks related to multimedia
DisplayPort Connection	categorical		The number of digital display interface standard primarily used to connect a computer to a monitor
HDMI Connection	categorical		The number of HDMI port found on dedicated graphics cards
Integrated	categorical		The card is built into the CPU or motherboard and shares system resources such as memory with the CPU

Table 2: *Variables Overview*

Variable	Data type	Unit	Description
L2 Cache	continuous	KB	A further on-chip cache for retaining copies of the data that travel back and forth between the SMs and main memory
Manufacturer	categorical		Company designes and manufactures graphics processing units
Max Power	continuous	Watts	Maximum power that system hardware can draw in a real-world scenario
Memory	continuous	MB	The on-chip memory available with GPUs for storing transient data buffers.
Memory Bandwidth	continuous	GB/sec	Measure how fast it can move data from/to memory (vRAM) to the computation cores
Memory Bus	continuous	Bit	The amount of data that can be transferred between the GPU and its memory at one time
Memory Speed	continuous	MHz	The speed at which data can be read from or written to the VRAM
Memory Type	categorical		A type of synchronous dynamic random-access memory (SDRAM)
Name	categorical		List of GPU
Notebook GPU	categorical		GTX graphics chip, but their internal numbers are still compelling
OpenGL	continuous		Open Graphics Library - A cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics
PSU	continuous	Watt & Amps	The part of a PC responsible for converting the alternating current (AC) power from an electrical outlet into direct current (DC) power that computer components can use
Pixel Rate	continuous	GPixel/s	The number of pixels that a GPU can render onto a screen every second
Power Connector	categorical	pin	The cable that connects the graphics card to the power supply unit, which provides the electricity needed for the card to function
Process	continuous	nm	The size of the transistors and other components within the chip
ROPs	categorical		Render Output Unit - a specific component on a GPU that is responsible for the processing of final pixel values prior to drawing them to the screen.
Release Date	continuous		The displayed date is either when the GPU was first announced or when that particular card was announced
Release Price	continuous		The specific price in currency on which GPU was officially introduced
Resolution WxH	categorical		Width and height dimension (display resolution) of an electronic visual display device, measured in pixels

Table 3: *Variables Overview*

Variable	Data type	Unit	Description
SLI Crossfire	categorical		The multi-GPU rendering methods
Shader	categorical		A computer program that calculates the appropriate levels of light, darkness, and color during the rendering of a 3D scene
TMUs	continuous		Texture mapping unit - low-level GPU component that operates with some independence, entirely dedicated to manipulating bitmaps and texture filtration
Texture Rate	continuous	GTexel/s	The number of textured pixels that a graphics card can render on the screen every second
VGA Connection	categorical		The number of standard display interface used to connect video output devices to computers and projectors to displays to monitors and TVs

Table 4: *Variables Overview*

2 Background

2.1 Multiple Linear Regression model

Multiple linear regression (MLR), also know simply as multiple regression, is a statistical technique that uses several explanatory variables to predict outcome of a response variable.
Assumption of MLR

- Homogeneity of variance: size of error in our prediction doesn't change significantly across the values of independent variable.
- Independence of observations: There must be no hidden relationships among variables. If two independent variables are too highly correlated ($r2 > 0.6$), then only one of them should be used in the regression model.
- Normality: The residuals, which are the differences between the observed values of the dependent variable and those predicted by the model, should be normally distributed.
- Linearity: There should be a linear relationship between the dependent variable and each of your independent variables. This can be checked using scatter plots or partial regression plots.

Model of MLR
The formula for a multiple linear regression is:

$$Y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon$$

Where, for $i = n$ observations:
Y: dependent variables
x_i: explanatory variables
β_0: y-intercept (constant term)
β_i: slope coefficients for corresponding explanatory variable x_i
ϵ: the model's error term (residuals)

2.2 Random Forest Regression Model

Random forest is a technique used in modeling predictions and behavior analysis and is built on decision trees. It can handle large data sets due to its capability to work with many variables running to thousands.

- Each tree is created from a different sample of rows and at each node, a different sample of features is selected for splitting.
- Each of the trees makes its own individual prediction.
- These predictions are then averaged to produce a single result.

Each decision tree has high variance, but when we combine all of them in parallel the the resultant variance is low, hence the output does not depend on one decision tree but on multiple decision trees.
Assumptions of Random forest model

- Input data: The input data shoulb be continuous, and the target variable is discrete. The input data contains multiple variables, and each variable has only one level.
- Missing values: There are no missing valus in the input data.
- Data distribution: The data is normally distributed.
- Correlation: The predictions from each tree must have very low correlations
- Non-linearity: Unlike linear regression, Random Forest does not assume a linear relationship between the dependent and independent variables.
- Error Distribution: Random Forest does not assume that the model errors are uncorrelated and uniform

When to use Random Forest model?

Random Forest Regression is often used to predict continuous values, such as predicting stock prices, time series forecasting, sale price predictions, etc. The random tree model works well with large datasets and captures non-linear relationships between input and target variables.

In addition, a random forest decision tree performs well with missing data and categorical variables, which makes it an ideal choice for many businesses that work with data containing varying feature types.

2.3 Q-Q plot

The quantile-quantile (q-q) plot is a graphical technique for determining if two data sets come from populations with a common distribution.

A q-q plot is a plot of the quantiles of the first data set against the quantiles of the second data set. By a quantile, we mean the fraction (or percent) of points below the given value. That is, the 0.3 (or 30%) quantile is the point at which 30% percent of the data fall below and 70% fall above that value.

The advantages of Q-Q plot are:

- Sample sizes do not need to be equal
- Many distributional aspects can be simultaneously tested

2.4 Box plot

Box plot uses boxes and lines to depict the distributions of one or more groups of numeric data. Box limits indicate the range of the central 50% of the data, with a central line marking the median value. Lines extend from each box to capture the range of the remaining data, with dots placed past the line edges to indicate outliers

Box plots are used to show distributions of numeric data values, especially when you want to compare them between multiple groups. They are built to provide high-level information at a glance, offering general information about a group of data's symmetry, skew, variance, and outliers

On the downside, a box plot's simplicity also sets limitations on the density of data that it can show. With a box plot, we miss out on the ability to observe the detailed shape of distribution

2.5 Histogram

In statistics, a histogram is a graphical representation of the distribution of data. The histogram is represented by a set of rectangles, adjacent to each other, where each bar represent a kind of data

We can use histogram when :

- Data has a single independent variable
- Data has a continuous range
- Two datasets need to be compared

The histogram looks similar to bar chart but there are differences between them:

Histogram	Bar Chart
It is a two-dimensional figure The frequency is shown by the area of each rectangle It shows rectangles touching each other	It is a one-dimensional figure The height shows the frequency and the width has no significance It consists of rectangles separated from each other with equal spaces

Table 5: *The difference between Histogram and Bar Chart*

2.6 Correlogram

A correlogram or correlation matrix allows to analyse the relationship between each pair of numeric variables of a dataset. The relationship between each pair of variable is visualised through a scatterplot, or a symbol that represents the correlation r_{XY} (bubble, line, number..)
The range of r_{XY}: -1$\leq r_{XY} \leq$1

- $-1 \leq r_{XY} < 0$: negative correlation. r_{XY} is closer to -1 indicating a stronger negative correlation between X and Y
- $0 < r_{XY} \leq 1$: positive correlation. r_{XY} is closer to 1 indicating a stronger positive correlation between X and Y
- r_{XY} is closer to 0 indicating a weak correlation between X and Y . $r_{XY} = 0$: indicating linearly independent between X and Y

Correlogram can help provide answers to the following questions:
1. Are the data random?
2. Is an observation related to an adjacent observation?
3. What is an appropriate model for the observed time series?

3 Data Proceeding

3.1 Feature Selection

In this assignment, we aim to analyze the memory speed of GPUs. Memory clock speed can have a significant impact on a GPU's performance, particularly for tasks that require the GPU to access a lot of data. A higher memory clock speed means that the GPU can access its memory faster, leading to better performance. We select some features that have the important effect on memory speed. There are 7 features that relate to our research: `"Architecture", "Core_Speed", "Memory", "Memory_Bandwidth", "Memory_Bus", "Memory_Type", "Process"`. [5][6]

3.2 Data Reading

We started by going through all the information provided in the dataset. We then displayed a preview of the first few lines to get a quick look at what the data looks like. This helped us get familiar with the dataset and understand its structure before diving deeper into our analysis. The dataset is shown in Figure 1.

```
> data <- read.csv("D:/Data/All_GPUs.csv", header = TRUE)
> head(data)
  Architecture Best_Resolution Boost_Clock Core_Speed DVI_Connection Dedicated Direct_X
1  Tesla G92b                              738 MHz              2       Yes  DX 10.0
2     R600 XT       1366 x 768                 \n-              2       Yes    DX 10
3    R600 PRO       1366 x 768                 \n-              2       Yes    DX 10
4       RV630       1024 x 768                 \n-              2       Yes    DX 10
5       RV630       1024 x 768                 \n-              2       Yes    DX 10
6       RV630       1024 x 768                 \n-              2       Yes    DX 10
  DisplayPort_Connection HDMI_Connection Integrated L2_Cache Manufacturer Max_Power   Memory
1                     NA               0          No      0KB       Nvidia 141 Watts 1024 MB
2                     NA               0          No      0KB          AMD 215 Watts  512 MB
3                     NA               0          No      0KB          AMD 200 Watts  512 MB
4                     NA               0          No      0KB          AMD            256 MB
5                     NA               0          No      0KB          AMD  45 Watts  256 MB
6                     NA               0          No      0KB          AMD  50 Watts  256 MB
  Memory_Bandwidth Memory_Bus Memory_Speed Memory_Type                          Name Notebook_GPU
1         64GB/sec   256 Bit      1000 MHz       GDDR3                GeForce GTS 150          No
2        106GB/sec   512 Bit       828 MHz       GDDR3          Radeon HD 2900 XT 512MB          No
3       51.2GB/sec   256 Bit       800 MHz       GDDR3                Radeon HD 2900 Pro          No
4       36.8GB/sec   128 Bit      1150 MHz       GDDR4 Radeon HD 2600 XT Diamond Edition          No
5       22.4GB/sec   128 Bit       700 MHz       GDDR3                 Radeon HD 2600 XT          No
6       35.2GB/sec   128 Bit      1100 MHz       GDDR4     Radeon HD 2600 XT 256MB GDDR4          No
  Open_GL                PSU  Pixel_Rate Power_Connector Process ROPs   Release_Date Release_Price
1     3.3 450 Watt & 38 Amps 12 GPixel/s            None     55nm   16 \n01-Mar-2009
2     3.1 550 Watt & 35 Amps 12 GPixel/s            None     80nm   16 \n14-May-2007
3     3.1 550 Watt & 35 Amps 10 GPixel/s            None     80nm   16 \n07-Dec-2007
4     3.3                     3 GPixel/s            None     65nm    4 \n01-Jul-2007
5     3.1 400 Watt & 25 Amps  3 GPixel/s            None     65nm    4 \n28-Jun-2007
6     3.3 400 Watt & 26 Amps  3 GPixel/s            None     65nm    4 \n26-Jun-2007
  Resolution_WxH SLI_Crossfire Shader TMUs Texture_Rate VGA_Connection
1      2560x1600           Yes      4   64  47 GTexel/s              0
2      2560x1600           Yes      4   16  12 GTexel/s              0
3      2560x1600           Yes      4   16  10 GTexel/s              0
4      2560x1600           Yes      4    8   7 GTexel/s              0
5      2560x1600           Yes      4    8   6 GTexel/s              0
6      2560x1600           Yes      4    8   6 GTexel/s              0
```

Figure 1: *Summary of data before cleaning*

However, in this assignment, we just select 8 features to analyze (1 main feature and 7 related features as mentioned in previous part), so that we just remove other features that have no relationship with `Memory_Speed` and show the preview of selected features in Figure 2

```
> head(selected_data)
  Architecture Core_Speed   Memory Memory_Bandwidth Memory_Speed Memory_Bus Memory_Type Process
1  Tesla G92b     738 MHz 1024 MB          64GB/sec     1000 MHz   256 Bit        GDDR3    55nm
2     R600 XT         \n-  512 MB         106GB/sec      828 MHz   512 Bit        GDDR3    80nm
3    R600 PRO         \n-  512 MB        51.2GB/sec      800 MHz   256 Bit        GDDR3    80nm
4       RV630         \n-  256 MB        36.8GB/sec     1150 MHz   128 Bit        GDDR4    65nm
5       RV630         \n-  256 MB        22.4GB/sec      700 MHz   128 Bit        GDDR3    65nm
6       RV630         \n-  256 MB        35.2GB/sec     1100 MHz   128 Bit        GDDR4    65nm
```

Figure 2: *Summary of selected data before cleaning*

3.3 Dealing with missing value

After choosing appropriate features, we now have the subset of original raw data set that contains merely 8 features. Nevertheless, There are some missing values, so that this cleaning process will handle **NA** values (missing values).

So, the first view of selected data before cleaning shown in Figure 3:

	Architecture	Core_Speed	Memory	Memory_Bandwidth	Memory_Speed	Memory_Bus	Memory_Type	Process
1	Tesla G92b	738 MHz	1024 MB	64GB/sec	1000 MHz	256 Bit	GDDR3	55nm
2	R600 XT	-	512 MB	106GB/sec	828 MHz	512 Bit	GDDR3	80nm
3	R600 PRO	-	512 MB	51.2GB/sec	800 MHz	256 Bit	GDDR3	80nm
4	RV630	-	256 MB	36.8GB/sec	1150 MHz	128 Bit	GDDR4	65nm
5	RV630	-	256 MB	22.4GB/sec	700 MHz	128 Bit	GDDR3	65nm
6	RV630	-	256 MB	35.2GB/sec	1100 MHz	128 Bit	GDDR4	65nm
7	R700 RV790 XT	870 MHz	2048 MB	134.4GB/sec	1050 MHz	256 Bit	GDDR5	55nm
8	R600 GT	-	256 MB	51.2GB/sec	800 MHz	256 Bit	GDDR3	80nm
9	Pitcairn XT GL	-	2048 MB	160GB/sec	1250 MHz	256 Bit	GDDR5	28nm
10	RV100	-	64 MB	2.9GB/sec	366 MHz	64 Bit	DDR	
11	NV28GL A2	-	128 MB	5.2GB/sec	325 MHz	128 Bit	DDR	150nm
12	Fermi GF110	650 MHz	6144 MB	177.6GB/sec	925 MHz	384 Bit	GDDR5	40nm
13	Kepler GK110	705 MHz	5120 MB	168GB/sec	1050 MHz	320 Bit	GDDR5	28nm
14	Kepler GK110	706 MHz	12288 MB	288.4GB/sec	1502 MHz	384 Bit	GDDR5	28nm
15	RV200	-	64 MB	5.8GB/sec	360 MHz	128 Bit	DDR	
16	GCN 1.1 Oland XT + Kaveri	1050 MHz	3072 MB	57.6GB/sec	900 MHz	128 Bit	DDR3	28nm
17	Kepler GK104 x2	-	8192 MB	320GB/sec	1250 MHz	512 Bit	GDDR5	28nm
18	Kepler GK110	732 MHz	6144 MB	249.6GB/sec	1300 MHz	384 Bit	GDDR5	28nm

Figure 3: *Selected data before cleaning*

After that, we calculated the number of missing data points to get an overview of the dataset's completeness. This allowed us to understand how much information was missing across the dataset, giving us insight into potential gaps or inconsistencies that might affect our analysis. The number and the probability of missing values are shown in Table 6.

Variable	Number of NA values	Probability of NA values
Architecture	12	0.0035
Core Speed	199	0.0584
Memory	85	0.0250
Memory Bandwidth	22	0.0065
Memory Speed	24	0.0070
Memory Bus	14	0.0041
Memory Type	8	0.0023
Process	95	0.0279

Table 6: *Summary of total NA values and Probability of NA values of each feature*

As depicted in Table 6, the proportion of missing data in each feature is relatively low (ranging from a minimum of 0.002348796 - `Memory_Type` to a maximum of 0.058426307 - `Core_Speed`). We observe that these ratios are small and unlikely to have a significant impact on the analysis results.

However, in this data set, it is difficult if the missing value does not have the same type. It may be blank, newline, dash,... So it is better to convert all of them into NA and the result is shown in Figure 4.
So, this is the **newData** file, where all missing value cells contain **NA**.

	Architecture	Core_Speed	Memory	Memory_Bandwidth	Memory_Speed	Memory_Bus	Memory_Type	Process
27	Fermi GF100	575 MHz	3072 MB	144GB/sec	750 MHz	384 Bit	GDDR5	40nm
28	Skylake GT3-E	300 MHz	*NA*	34.1GB/sec	1067 MHz	128 Bit	DDR4	14nm
29	Skylake GT3-E	300 MHz	*NA*	34.1GB/sec	1067 MHz	128 Bit	DDR4	14nm
30	Haswell GT3	200 MHz	*NA*	25.6GB/sec	800 MHz	128 Bit	DDR3	22nm
31	NV18 A4	*NA*	64 MB	3.2GB/sec	200 MHz	128 Bit	DDR	150nm
32	Broadwell GT3-E	300 MHz	*NA*	29.9GB/sec	933 MHz	128 Bit	DDR3	14nm
33	NV17 A3	*NA*	64 MB	3.2GB/sec	200 MHz	128 Bit	DDR	150nm
34	NV18 A4	*NA*	128 MB	4.1GB/sec	256 MHz	128 Bit	DDR	150nm
35	NV25 A2	*NA*	64 MB	3.6GB/sec	225 MHz	128 Bit	DDR	150nm
36	Skylake GT3-E	300 MHz	*NA*	34.1GB/sec	1067 MHz	128 Bit	DDR4	14nm
37	Skylake GT3-E	300 MHz	*NA*	34.1GB/sec	1067 MHz	128 Bit	DDR4	14nm
38	NV17 A3	*NA*	128 MB	2.7GB/sec	166 MHz	128 Bit	DDR	150nm

Figure 4: *Convert missing values into "NA"*

Cleaning procedure retains NA values unless they are deemed necessary to remove. This is because certain instances may contain vital information that should not be discarded. Depending on the context of the study, NA values cannot be universally considered as invalid data. In subsequent sections, when we concentrate on specific data patterns, the NA cleaning will be customized to ensure no crucial instances are inadvertently lost.

Based on the probability of NA values in each feature, the probability of NA values of all features are small, so to easily handle this data, we can delete all rows that have NA values. The result is shown in Figure 5.

	Architecture	Core_Speed	Memory	Memory_Bandwidth	Memory_Speed	Memory_Bus	Memory_Type	Process
1	Tesla G92b	738 MHz	1024 MB	64GB/sec	1000 MHz	256 Bit	GDDR3	55nm
7	R700 RV790 XT	870 MHz	2048 MB	134.4GB/sec	1050 MHz	256 Bit	GDDR5	55nm
12	Fermi GF110	650 MHz	6144 MB	177.6GB/sec	925 MHz	384 Bit	GDDR5	40nm
13	Kepler GK110	705 MHz	5120 MB	168GB/sec	1050 MHz	320 Bit	GDDR5	28nm
14	Kepler GK110	706 MHz	12288 MB	288.4GB/sec	1502 MHz	384 Bit	GDDR5	28nm
16	GCN 1.1 Oland XT + Kaveri	1050 MHz	3072 MB	57.6GB/sec	900 MHz	128 Bit	DDR3	28nm
18	Kepler GK110	732 MHz	6144 MB	249.6GB/sec	1300 MHz	384 Bit	GDDR5	28nm
21	Fermi GF100	575 MHz	6144 MB	144GB/sec	750 MHz	384 Bit	GDDR5	40nm
22	Fermi GF100	575 MHz	6144 MB	144GB/sec	750 MHz	384 Bit	GDDR5	40nm
27	Fermi GF100	575 MHz	3072 MB	144GB/sec	750 MHz	384 Bit	GDDR5	40nm
47	Kepler GK110-400-A1	837 MHz	6144 MB	288.4GB/sec	1502 MHz	384 Bit	GDDR5	28nm
48	Kepler GK110-430-B1 (x2)	705 MHz	12288 MB	672GB/sec	1750 MHz	384 Bit	GDDR5	28nm

Figure 5: *Data after deleting all rows that have NA values*

3.4 Dealing with features' unit

We noticed that all the data within one feature shares the same unit. Therefore, to facilitate easier observation during the analysis process, we decided to remove all units from the `Core Speed`, `Memory`, `Memory Bandwidth`, `Memory Bus`, `Process` features. This uniformity will streamline our analysis and make it simpler to compare and interpret the data across these features. Figure 6 show the data after cleaning process. This table of data can be used to analyze in the following process.

	Architecture	Core_Speed	Memory	Memory_Bandwidth	Memory_Speed	Memory_Bus	Memory_Type	Process
1	Tesla G92b	738	1024	64.0	256	256	GDDR3	55
7	R700 RV790 XT	870	2048	134.4	256	256	GDDR5	55
12	Fermi GF110	650	6144	177.6	384	384	GDDR5	40
13	Kepler GK110	705	5120	168.0	320	320	GDDR5	28
14	Kepler GK110	706	12288	288.4	384	384	GDDR5	28
16	GCN 1.1 Oland XT + Kaveri	1050	3072	57.6	128	128	DDR3	28
18	Kepler GK110	732	6144	249.6	384	384	GDDR5	28
21	Fermi GF100	575	6144	144.0	384	384	GDDR5	40
22	Fermi GF100	575	6144	144.0	384	384	GDDR5	40
27	Fermi GF100	575	3072	144.0	384	384	GDDR5	40
47	Kepler GK110-400-A1	837	6144	288.4	384	384	GDDR5	28

Figure 6: *Data after removing units*

Additionally, we can recheck the number of missing values to confirm that this is indeed the finalized dataset after cleaning. As illustrated in Figure 7, there are no missing data points present in the final dataset. This confirms that our data cleaning process has been successful, and we now have a complete dataset ready for analysis.

```
> apply(is.na(newData), 2, sum)
    Architecture       Core_Speed           Memory Memory_Bandwidth
               0                0                0                0
    Memory_Speed       Memory_Bus      Memory_Type          Process
               0                0                0                0
```

Figure 7: *Rechecking NA values*

Finally, after the cleaning data step, we now have a summary of new data file by using **summary()** in R. Then, we have the summary of the data in Figure 8.

```
> summary(newData)
 Architecture         Core_Speed        Memory      Memory_Bandwidth
 Length:2333        Min.   : 200.0   Min.   :  128   Min.   :   4.0
 Class :character   1st Qu.: 810.0   1st Qu.: 1024   1st Qu.:  72.0
 Mode  :character   Median : 993.0   Median : 2048   Median : 134.4
                    Mean   : 976.6   Mean   : 3179   Mean   : 165.7
                    3rd Qu.:1100.0   3rd Qu.: 4096   3rd Qu.: 224.4
                    Max.   :1784.0   Max.   :24576   Max.   :1280.0
  Memory_Speed      Memory_Bus     Memory_Type          Process
 Min.   :  64.0   Min.   :  64.0   Length:2333        Min.   :14.00
 1st Qu.: 128.0   1st Qu.: 128.0   Class :character   1st Qu.:28.00
 Median : 192.0   Median : 192.0   Mode  :character   Median :28.00
 Mean   : 225.1   Mean   : 225.1                      Mean   :30.44
 3rd Qu.: 256.0   3rd Qu.: 256.0                      3rd Qu.:40.00
 Max.   :4096.0   Max.   :4096.0                      Max.   :90.00
```

Figure 8: *Data Summary*

3.5 Encoding categorical variable

In the dataset, we have two variables that have categorical data type. Therefore, we need to encode two data variables: Memory_Type and Architecture. We use the **factor()** function to convert the data columns to factor, then use the **as.numeric()** function to convert the factor to numeric data. Some of the results are shown in Figure 9.

Memory_Type_Num	Architecture_Num
1	1
2	2
2	3
2	4
2	4
3	5
2	4
2	6
2	6

Figure 9: *Character variables in numeric form*

4 Descriptive Statistics

4.1 Box plot and Histogram

Based on cleaned data set in **section 3**, we can plot the histograms and boxplots of predictor variables by using the **boxplot()** and **hist()** function. Since the data is taken form Kaggle (A reputable data site about GPUs), we will still get outliers without removing them.

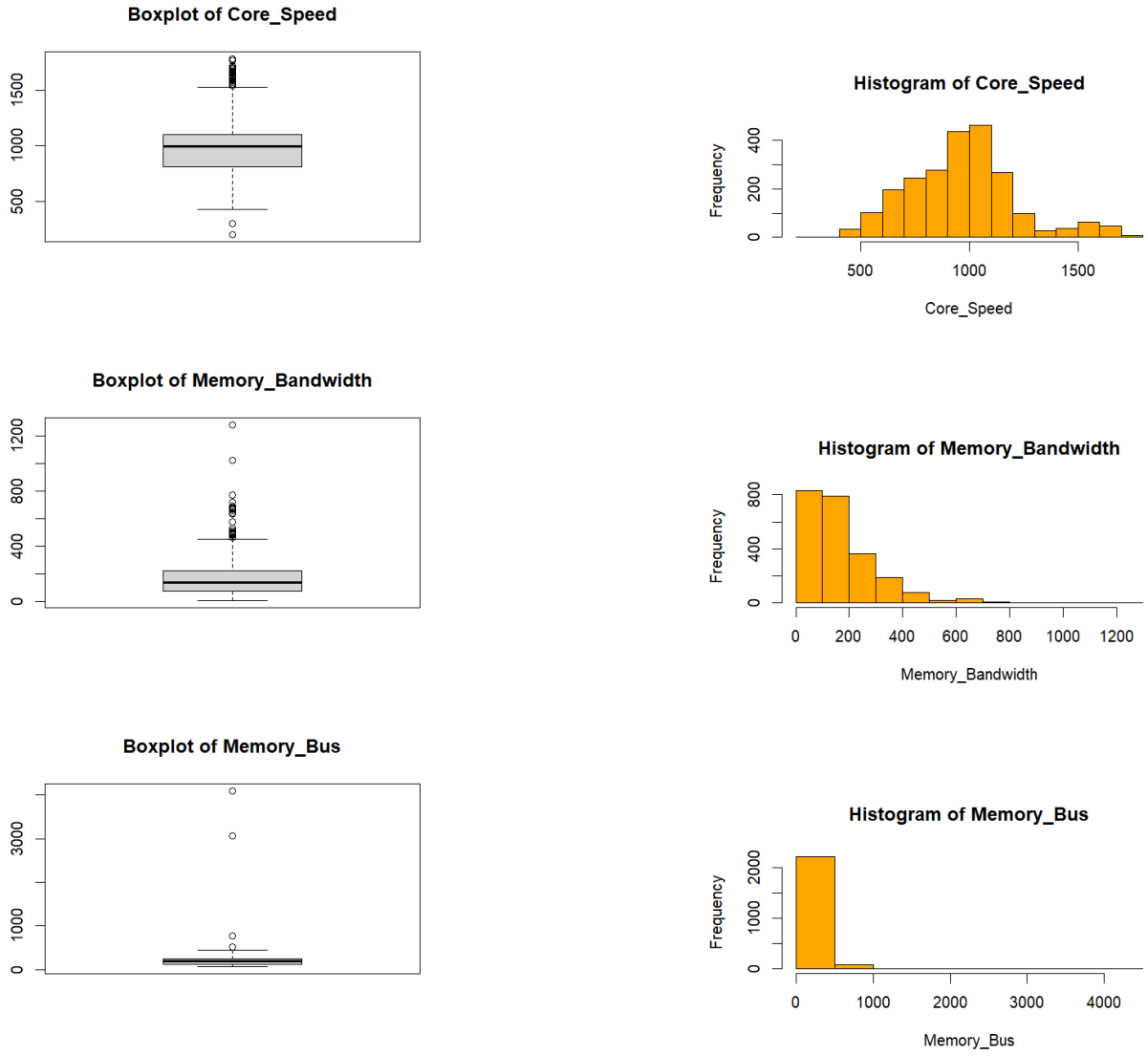


Figure 10: *Box plot and Histogram of Core Speed, Memory Bandwidth and Memory Bus*

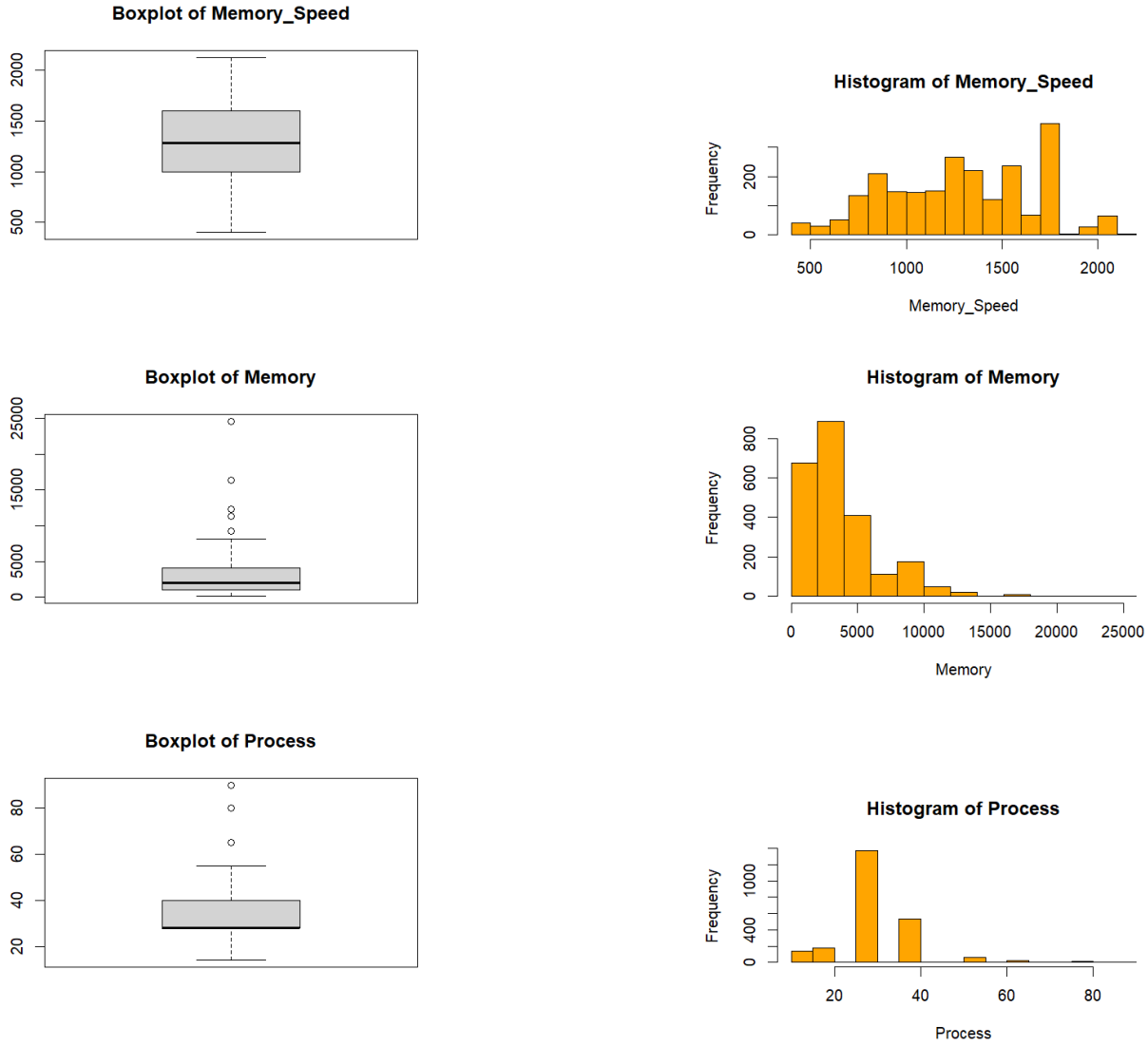


Figure 11: *Box plot and Histogram of Memory speed, Memory and Process*

According to the summary and plots of Figure 10 and Figure 11 above:

- **Core_Speed**: core_speed ranges from 200 to 1784. The median of core_speed is 993 which is more than its means (976.6), so the distribution is left-skewed
- **Memory_Bandwidth**: Memory_Bandwidth ranges from 4 to 1280. As a mean of 165.7 which is greater than the median, the distribution is right-skewed. This suggested that some GPUs can typically have high memory_bandwidth
- **Memory**: Memory has a large range from 128 to 24576. With a mean of 3179, which is larger than the median 2048, the distribution is also right-skewed. Some GPUs have exceptionally high memory
- **Memory_Speed**: This variable ranges from 400 to 2127. The median of memory_speed is 1280 which is little less than its means (1303), so the distribution is right-skewed. This indicates that we can have high memory_speed
- **Memory_Bus and Process**: These variables also have right-skewed, with means are larger than their medians

4.2 Correlation matrix

Based on above information, we can have a correlation plot to have better view of relation between variables. Then we will have the result in Figure 12:

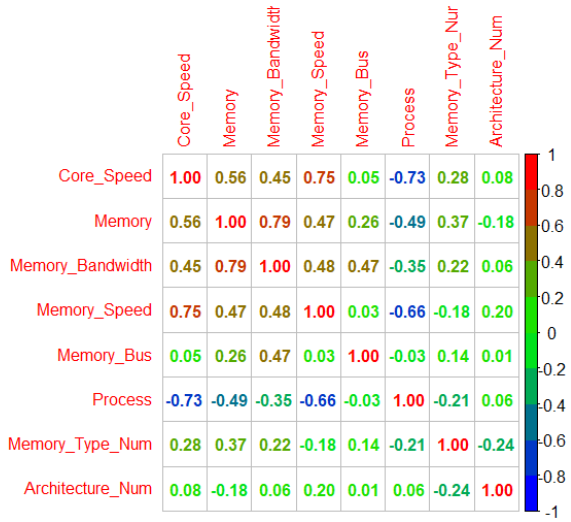


Figure 12: *Correlation matrix of observed dataset using Coefficient of Determination*

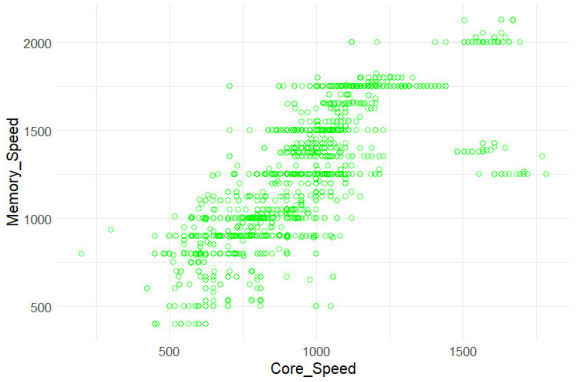
According to Figure 12, we can see:

- **Memory_Speed**: This variable has a strong positive correlation with Core_Speed, Memory and Memory_Bandwidth, which means the speed of memory are highly dependent on these variables. While other variables show weak negligible with this variable so that the memory_speed is not strongly influenced by these factors
- **Core_Speed**: As Core_Speed increases, the processing power of the core rises, leading to a positive correlation with Memory, Memory_Bandwidth, and Memory_Speed. This is because higher core_speed demands faster data access and transfer rates from memory to keep pace with increased computational demands
- **Memory**: It shows a high positive correlation with Core_Speed, Memory_Bandwidth, Memory_Speed and Memory_Type. This indicates that the capacity of memory can strongly effect the Memory_Bandwidth and have significant relation with other parts of CPU
- **Memory_Bandwidth**: It represents the rate at which data can be transferred to and from memory, exhibits a positive correlation with Memory, Core_Speed, Memory_speed, and Memory_Bus. A higher Memory_Bandwidth ensures efficient data transfer between memory and the core, which becomes increasingly important as Memory and Memory_Bus operate at faster rates, requiring a higher bandwidth to support optimal performance. Moreover, the correlation between memory bandwidth and memory is 0.79, which indicates that these 2 factors have very strong dependence. So we just choose memory bandwidth for our model and eliminate memory factor.
- **Process and Architecture**: have slightly positive correlation to each other and both of them are not affected by other variables
- **Memory_Type**: This variable just has positive correlation with Core_Speed, Memory, Memory_Bandwidth and Memory_Bus, which means that these factors do not strongly influence to Memory_Type

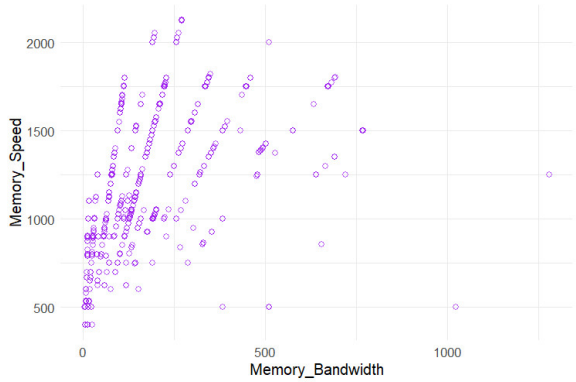
4.3 Variable trend

We plot scatter plots and box plot to know, if we were given a particular value of memory bandwidth, memory bus, memory, core speed, ... what a good prediction would be for memory speed.

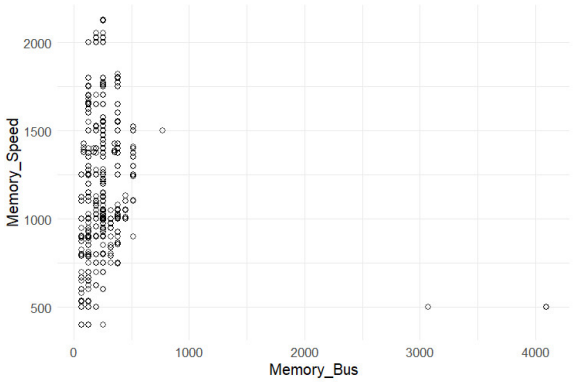
(a) Scatter plot of Architecture vs Memory Speed



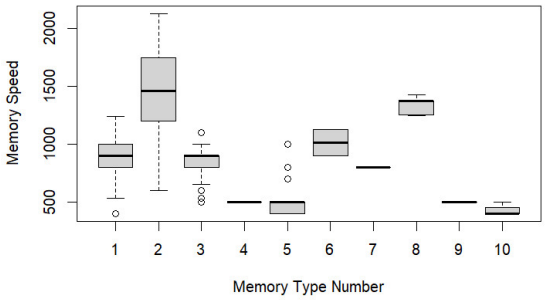
(b) Scatter plot of Core Speed vs Memory Speed



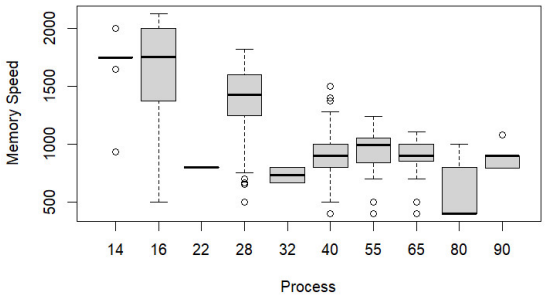
(c) Scatter plot of Memory Bandwidth vs Memory Speed



(d) Scatter plot of Memory Bus vs Memory Speed



(e) Box plot of Memory Type vs Memory Speed



(f) Box plot of Process vs Memory Speed

Figure 13: *Plots illustrate relationship between memory speed and different types of architecture, core speed, memory bandwidth, memory bus, memory type, process respectively*

- **Architecture types vs Memory speed - Figure 13.a**: A noticeable concentration of data points exists between 1000 and 1500 Memory speed and less than 100 Architecture_Num, which could represent a common performance range for many GPU types. The data is quite spread out, indicating a wide range of performance across different GPU architectures. We can see that the relationship between Architecture types and Memory speed is not strictly linear so we may use Polynomial Regression, Random forest regression,... for further prediction
- **Core speed vs Memory Speed - Figure 13.b**: Data concentration of Core speed is around 750 to

1250 and Memory speed is about 1000 to 1500. Some outliers are present, particularly at higher Core Speeds where Memory Speeds do not correspondingly increase. The plot shows a positive correlation, indicating that as Core speed increases, Memory speed tends to increase, suggesting that faster cores are often paired with faster memory. To model the relationship between core speed and memory speed as depicted in the scatter plot, we should consider some approached such as Linear Regression, Polynomial Regression,...

- **Memory bandwidth vs Memory Speed - Figure 13.c**: Most data points are clustered in the lower left corner, showing that lower memory speeds and bandwidths are more common. Despite some variability, there is a general upward trend, meaning that increases in memory bandwidth tend to accompany increases in memory speed.
- **Size of Memory bus vs Memory speed - Figure 13.d**: Most data points are clustered between 0 to 1000 on the Memory_bus axis and between 500 to 1500 on the Memory_Speed axis, indicating common configurations for memory speed and bus size. There are 2 significant outliers at around 3000 and 4000 whereas all others points lie with in the range [0,1000]. The lack of a clear trend suggests that the relationship between memory speed and bus size is not straightforward and may be influenced by other factors.
- **Memory Type and Memory speed - Figure 13.e**: A significant concentration of data points at the Memory type 2 with memory speeds mostly above 1000, indicating that this memory type is associated with higher speeds. Other Memory type show variability with fewer and generally lower speed data points, except for type 8 which also has some higher speed instances.
- **Process vs Memory speed - Figure 13.f**: There is no clear upward and downward trend. To predict memory speed, we would typically need more data points and possibly employ statistical models like regression analysis.

5 Multiple Linear Regression model in analyzing and predicting memory speed of GPUs

Memory Speed is an interesting characteristic that comes with the GPU. Higher memory speeds generally result in improved performance and responsiveness, especially in tasks that require rapid data access, such as gaming and graphics rendering.

As stated in our Report's title, the main focus of the analysis will be *Memory Speed* and the affect of other attributes (variables) on this dependent variable.

We notice that as we have analyzed and selected in the previous section, the predictors are `"Architecture"`, `"Core_Speed"`, `"Memory_Bandwidth"`, `"Memory_Bus"`, `"Memory_Type"`, `"Process"`.

5.1 Data Splitting

First, we load the cleaned data set from the cleaning process above. Then we do data splitting to split the dataset into two subsets: training set and test set. The train set randomly get 70% while the test set get the other 30% of the observations.

Data splitting is an important aspect of data analysis, particularly for creating models based on data.

5.2 Multiple Linear Regression Analysis

5.2.1 Model Purpose

In this section, we will investigate the relationship between **Memory_Speed** and other features of GPUs. Indeed, there are many regression approaches such as (Simple or Multiple) Linear Regression, Polynomial Regression, Logistic Regression, Random Forrest Regression Model, and more. In this section, we are looking forward to using **Multiple Linear Regression**

5.2.2 Model Definition

The formula for Multiple Linear Regression model is:

$$Y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon$$

We use the least squared criterion to fit multiple linear regression model:

$$\widehat{Y} = \widehat{\beta_0} + \widehat{\beta_1} x_1 + ... + \widehat{\beta_n} x_n$$

Where, for $n = 6$ predictors:
$\widehat{Y}$: dependent, target variables - memory speed feature
x_i: explanatory, predictor variables - Architecture, Core_Speed, Memory_Bandwidth, Memory_Bus, Memory_Type, Process
$\widehat{\beta_0}$: y-intercept (constant term)
$\widehat{\beta_i}$: slope coefficients for corresponding explanatory variable x_i
$e_i = y$: the model's error term (residuals) We perform hypothesis tests for each predictor:

$$H_0 : \widehat{\beta_i} = 0, i \in 1, 2, ..., 6$$
$$H_1 : \widehat{\beta_i} \neq 0, i \in 1, 2, ..., 6$$

- **Null Hypothesis H_0:** The slope coefficient of the predictor variable is equal to 0. That means there is no relationship between the target variable and the predictor variable.
- **Alternative Hypothesis H_1:** The slope coefficient of the predictor variable is not equal to 0, implying that there is a significant relationship between the target and the predictor.

There are some metrics that associate with the hypothesis: t-value and p-value

- **t-value:** a way to quantify the difference between the population means. A higher absolute t-value indicates that the predictor has more significant influential in analyzing and predicting "memory speed".
- **p-value:** the probability of obtaining a t-value with an absolute value at least as large as the one we actually observed in the sample data if the null hypothesis is actually true. If the p-value is less than 0.05 (5%) then we reject the null hypothesis of the test, indicate that has statistical significance.

5.2.3 Model Fitting

Firstly, we will have Linear Regression for feature **Memory_Speed** with other features by using `lm()` function in R-code. Then we have the summary table in Figure 14:

```
Call:
lm(formula = Memory_Speed ~ Architecture + Core_Speed + Memory_Bandwidth +
    Memory_Bus + Memory_Type + Process, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-618.70 -111.94   -0.11  114.29  860.59

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      945.49640   38.39026  24.629  < 2e-16 ***
Architecture       0.15518    0.04369   3.552 0.000393 ***
Core_Speed         0.85016    0.02628  32.355  < 2e-16 ***
Memory_Bandwidth   0.72149    0.03872  18.633  < 2e-16 ***
Memory_Bus        -0.10206    0.01844  -5.536 3.61e-08 ***
Memory_Type     -131.60271    3.72889 -35.293  < 2e-16 ***
Process           -9.63628    0.58188 -16.561  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 164.6 on 1626 degrees of freedom
Multiple R-squared:  0.8057,    Adjusted R-squared:  0.805
F-statistic:  1124 on 6 and 1626 DF,  p-value: < 2.2e-16
```

Figure 14: *Summary of Multiple Linear Regression model*

The figure show that all of predictors have p-value less than 0.05 and all of coefficient estimates are different from 0. Therefore, we can reject the null hypothesis. That means predictors variables have significant relationship with memory speed.

5.2.4 Model Conclusion

5.2.4.a Model Fitting Interpretation

Here are some notable data in Figure 14:

- **Residual:** indicates the difference between the observed value and the fitted value.
  - The residual is range from -618.70 to 860.59. This means some predictions are quite far from the actual values.
  - Residual standard error: assess how well a linear regression model fits the data. The smaller the residual standard error, the better a regression model fits a dataset. This value is 164.6 on 1626 degrees of freedom. It means that this model quite fit the dataset, not much.
- **Coefficients:** indicate the level of significance of each predictor in predicting "memory speed". Variables with higher absolute t-value (lower p-value) are more significant predictors. In Figure 14, it shows that all predictors have p-value lower than 0.05. That means all of them are crucial predictors.
- **Multiple R-squared:** the proportion of the variation in dependent variable that can be explained by the independent variables. In this model, this value means approximately 80.57% of the variability in Memory Speed is explained by the predictors of the model and the rest 19.43% is defined by other factors. This is a reasonably good fit.
- **Adjust R-squared:** a modified version of R-squared that has been adjusted for the number of predictors in the model. This value is 80.5%, very close to multiple R-squared. This indicates that adding additional predictors would not greatly enhance the model's performance.

- **F-statistic:** This value test the significance of the model. A large F-statistic value proves that the regression model is effective in its explanation of the variation in the dependent variable and vice versa.

5.2.4.b Features Selection

Based on the model fitting, we get a conclusion that the most significant factors that affect the memory speed are architecture, core speed, memory bandwidth, memory bus, memory type and process. So that, the customers can consider those factors when choosing an appropriate GPU.

5.2.4.c Effects of Predictors on Memory Speed

By analyzing the parameter estimates in the summary table (Figure 14), we interpret the effects of the predictors on memory speed. This value describes the size of the contribution of that predictor; a near-zero coefficient indicates that variable has little influence on the response. The sign of the coefficient indicates the direction of the relationship [13]

For a numerical predictor variable: the regression coefficient represents the difference in the predicted value of the response variable for each one-unit change in the predictor variable.

- A positive estimate indicates that as the value of the predictor increases, the mean of the target variable also tends to increase (core speed, memory bandwidth).
  - Core speed: estimate = 0.85016. It means if the core speed increases 1 MHz, the memory speed tends to increases 0.85016 MHz
  - Memory bandwidth: estimate = 0.72149. It means if the memory bandwidth increases 1 GB/sec, the memory speed tends to increase 0.72149 MHz
- A negative estimate suggests that as the predictor increases, the target tends to decrease (memory bus, process)
  - Memory bus: estimate = -0.10206. It means if the memory bus increases 1 Bit, the memory speed tends to decrease 0.10206 MHz.
  - Process: estimate = -9.63628. It means if the process increases 1 mn, the memory speed tends to decrease 9.63628 MHz.

For a categorical predictor variable: the regression coefficient represents the difference in the predicted value of the response variable between each predictor variable. The value of predictors range from 1 to numbers of group of that predictors. (architecture, memory type).

- Architecture: estimate = 0.15518. It means that the memory speed will increase 0.15518 $\times$ type of architecture (the type which has encoded by factor in section 4)
- Memory Type: estimate = -131.60271. It means that the memory speed will decrease 131.60271 $\times$ type of memory (the type which has encoded by factor in section 4)

The standard error (SE) indicate the variation of the observed data compared to the fitted linear regression line. A small SE is an indication that the sample mean is a more accurate reflection of the actual population mean. In figure 14, it indicates that the SE of all predictors and intercept are small, which suggest the sample mean closely approximates the true population mean.

5.2.5 Checking Model Assumptions

Now we need to check for Multiple Linear Regression model assumptions:

- **Homogeneity of variance:** The variance of error terms (residuals) should be consistent across all levels of the independent variables. We use ***Scale-Location plot*** to test for Residual Errors have Constant Variance. By using this plot, we can see the fitted values and the square root of the standardlized residuals. We expect that the red line is roughly horizontal across the plot and the spread of the residuals is roughly equal at all fitted values.

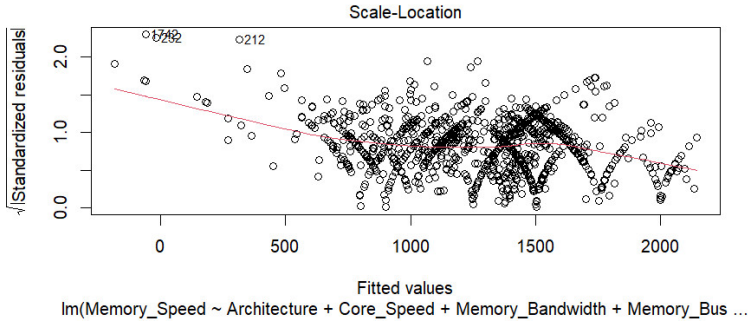


Figure 15: *Scale-Location plot to check for Homogeneity of variance*

In the scale-location plot above, we can see that:

  - The red line is roughly horizontal across the plot around value 1.
  - There is no clear pattern among the residuals. It is randomly scattered around the red line with roughly equal variability at all fitted values.

  Therefore, it is implied that the assumption of Homogeneity of variance is met.

- **Normality:** The residuals (the differences between observed and predicted values) are normally distributed. We use ***Q-Q plot*** to test for normality of the errors. We expect that the residual will mostly lie on or close to a 45-degree line (y = x line).

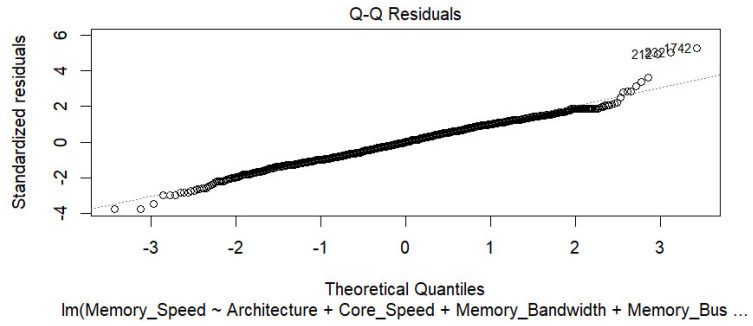


Figure 16: *Q-Q plot to check for normality*

In the above plot, it can be clearly seen that, most of points are lying on or near the line y = x, whereas merely few points at 2 tails are not too near the line. Therefore, we can conclude that the data follows normal distribution.

After summary the model, we can conclude that this model is quite accurate, after observing 2 plots and the summary table. The prediction of this model will be presented later in Section 7 below.

6 Discussion and Extension

6.1 Discussion

After conducting and experiment with Multiple Linear Regression(MLR) to analyze the impact of some GPU's characteristics on memory speed and making some predictions, we can conclude some **advantages of MLR**:

- Simplicity and Interpretability: Linear Regression is straightforward to understand and implement. The relationship between independent and dependent variables is expressed as a linear equation, making the model results easy to interpret
- Computationally Efficient: It is less computationally intensive, making it a good choice for very large datasets where model training speed is a concern
- Less Data Needed: It can perform well with a smaller amount of data and does not require as much data to reach convergence

However, we have also encountered several **disadvantages** while using this Multiple Linear Regression model:

- Assumption-Heavy: Linear Regression assumes a linear relationship between the dependent and independent variables. It also assumes normality, homoscedasticity (constant variance of residuals), and independence of errors. We have devoted a considerable amount of time to conducting tests for assumptions, which has proven to be both complex and exhaustive.
- Sensitive to Outliers: While performing data cleaning, we did not remove the outliers which can have a disproportionately large effect on the fit on our model. This problem can result to large errors and low accuracy

6.2 Random Forrest Regression Model

We may consider utilizing an alternative regression model such as the **Random Forest**. This is a robust machine learning model capable of handling complex and non-linear relationships between independent and dependent variables.

A Random forest regression model combines multiple decision trees to create a single model. Each tree in the forest builds from a different subset of the data and makes its own independent prediction. The final prediction for input is based on the average or weighted average of all the individual trees' predictions.

This algorithm can handle both continuous and categorical data, provides useful insights through feature importance scores, which help understand which variables are influencing the outcome the most. Moreover, Random Forest is less sensitive to outliers than Linear Regression because it uses multiple decision trees to make predictions. Therefore, it is suitable for modeling GPU attributes data since it contain a mix of numerical and categorical variables.

Over fitting occurs when a machine learning model performs exceptionally well on the training data but fails to accurately predict new or test data. In the case of the **Random Forest Regression model**, this can happen when there are too many trees in the forest or when the trees are allowed to grow without depth restrictions. There are several techniques to prevent over fitting, including **limiting tree depth**, **increasing the minimum number of samples at leaf nodes, using bootstrap samples,...**

In this model, we will analyze and predict the target variable "memory speed" in the relationship with all the feature we selected before. They are "Architecture", "Core_Speed", "Memory", "Memory_Bandwidth", "Memory_Bus", "Memory_Type", "Process". The Figure 17 below shows the summary of the model.

```
Call:
 randomForest(formula = Memory_Speed ~ Architecture + Core_Speed +
Memory_Bandwidth + Memory_Bus + Memory_Type + Process, data = train_dat
a)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 3227.691
                    % Var explained: 97.67
```

Figure 17: *Summary of Random Forest Regression*

Explanation:

- **Mean of squared residuals**: the MSE (mean squared error) of the out-of-bag errors. In the figure below, it shows the error is related to the number of trees. At the beginning, the number of trees is small which leads to the very high error (around 11000). When the number of trees increase, the error decrease. At the point of around 30 trees, the error is saturated. It means that if we increase trees after this point, it will not improve the error anymore. That is the optimal point. The Figure 18 shows the error related to the number of trees.

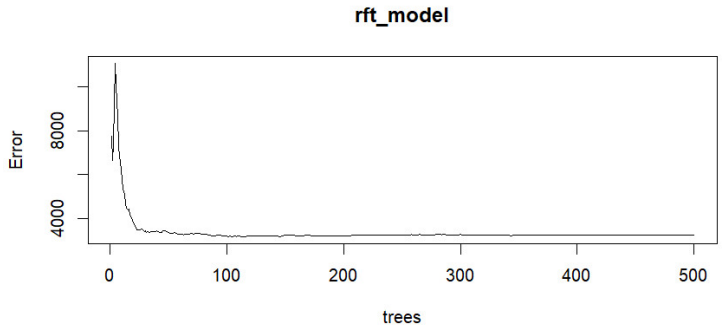


Figure 18: *Plot of the number of trees by the mean squared error.*

- **% Var explained:** The goodness of the out-of-bag prediction explained targeted variance in the training dataset. This value is 97.7%, which indicates that this model is pretty good.

Checking for assumption: The random forest regression requires the the residual must follow normal distribution. Therefore, we use Q-Q plot to check for the normality of the model as the linear regression model.

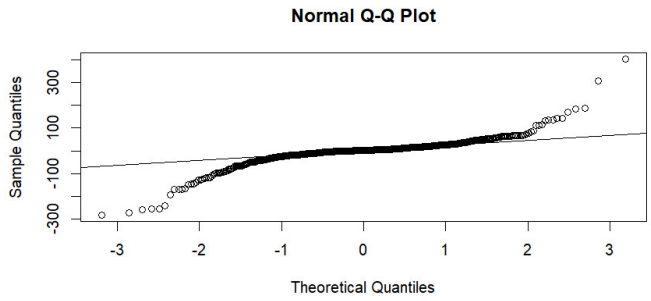


Figure 19: *Q-Q plot for checking normality assumption of Random Forest*

The figure show that almost points are lying on or close to the y = x line, except a small amount of point in 2 tails are quite far from the line. It still indicated that the model is normally accurate.

7 Model Prediction

In this section, we will compare Multiple Linear Regression model and Random Forest Regression Model, then we use the previous test set to calculate the accuracy of the Multiple Linear Regression and Random Forest Regression models.

7.1 Model Comparison

We analyze of some criterion include: Accuracy, Mean Absolute Error, R-squared to find which model is fitter in with our dataset.

Criterion \ Model	Multiple Linear Regression	Random Forest Regression
Mean Absolute Error	127.087799	28.599910
R-squared	0.8057	0.9818
Accuracy	88.3160%	97.3067%

Table 7: *Mean Absolute Error, R-squared, Accuracy of Multiple Linear Regression and Random Forest Regression*

In Table 7, it is easy to see that the MAE, R-squared and Accuracy of Random Forest Regression are higher than those of Multiple Linear Regression.

- Mean Absolute Error: The Random Forest Regression has MEA that is smaller than Multiple Linear Regression more than 4 times. It means that the difference between the measured value and "true" value of Random Forest Regression is small and it is a really good error when choosing model to fit the data.
- R-Squared: This is the coefficient of determination. The Random Forest Regression has R-squared value that is higher than Multiple Linear Regression. This value is very close to 1, so that Random Forest Regression has a good fit to our data.
- Accuracy: The accuracy of Random Forest Regression is also higher than the Multiple Linear Regression model. This value of Random Forest Regression is 97.3076%, which is very close to absolutely accurate.

Based on these values, we can conclude that the Random Forest Regression is more reliable than Multiple Linear Regression model. It has a very good error, suitability and accuracy, so that we can predict the value of memory speed by using Random Forest Regression model to have better results.

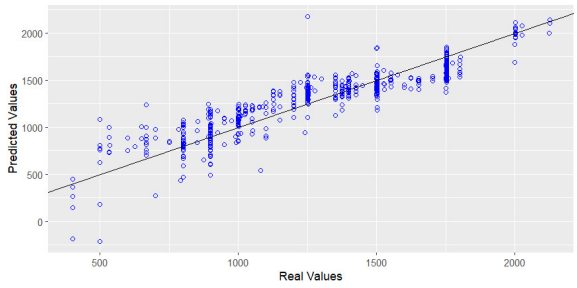
7.2 Model Testing

In this section, we will compare the predicted value using 2 models and the real values of the test set, which we have splitted before.

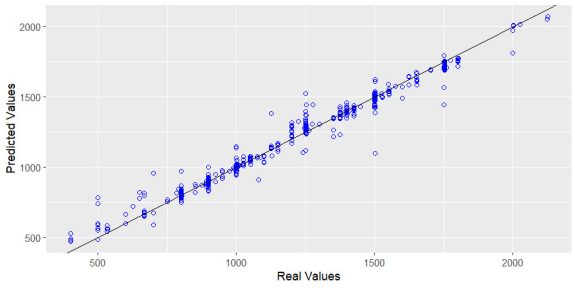
Real Values	Predicted Values	
	Multiple Linear Regression	Random Forest Regression
925	938.8553	893.7064
1502	1182.1962	1434.4994
1753	1818.6757	1725.5703
1502	1502.7319	1518.0371
500	1080.4414	740.4445
1502	1294.0331	1481.1528
1750	1373.5157	1442.1008
993	837.6465	982.6096
1000	859.1891	984.4606
990	901.2644	993.9300
900	790.9646	907.6867
800	929.8748	811.8967
700	273.9411	586.9225
800	929.8748	811.8967
400	269.0846	471.8451
800	800.1129	762.5174
800	929.8748	811.8967
900	855.5027	918.5370
500	183.8574	565.2910
1200	1224.4617	1220.3814

Table 8: *Real Values and Predicted Values of Multiple Linear Regression and Random Forest Regression Models*

Table 8 shows some observations of real value and associated predicted value of Multiple Linear Regression and Random Forest Regression. We can see that some predicted values of MLR are very far from the observed values. Most of the predicted values differed by more than 100 units from the actual values. However, the Random Forest Regression generates predicted values that are very close to the observed. The difference between them mostly less than 100 units.



(a) Multiple Linear Regression



(b) Random Forest Regression

(c) Combination of 2 models

Figure 20: *Scatter plot of predict values versus real values of memory speed using 2 models*

Using the plot, we will have a better visualization of the predicted values. We expect that the predicted values should be close to the line y = x. This line means that the predicted values are equal to the real values. The Figure 20 indicates that the predicted values of Multiple Linear Regression and Random Forest Regression is scattered around the line y = x. However, the dots of Random Forest is closer to y = x than those of Multiple Linear Regression.

In conclusion, compared to Multiple Linear Regression model, the Random Forest Regression model gives more optimal results which are proven by the error, accuracy, R-squared values. It seem to be fitter to our data and get a better result than the Multiple Linear Regression.

8 Conclusion

In summary, we have covered an overview of the dataset, the theoretical foundation, and used several models to predict and produce positive results for predicting memory speed based on certain features of GPUs. We've carefully selected features that have a correlation and influence on the changes in memory speed. They are architecture, core speed, memory bandwidth, memory bus, memory type and process. This helps the producers and consumers to predict the memory speed base on those values, produce and choosing appropriate one that is suitable for the demand.

By using R programming language, we've been able to better understand how to gather, process, and analyze data over the long term, and even make educated guesses about future trends. Of course, when putting our ideas into practice, there might be some small mistakes, and our predictions might not always match up perfectly with reality. After analyzing Multiple Linear Regression and Random Forest Regression, we have a conclusion that the Random Forest Regression provides better result and more reliable. In reality, we can use this model to dealing with memory speed of GPUs.

9 Data and Code availability

9.1 Data source

Link: Data Set Link

9.2 Code source

Link: Source Code Link

```
#####LIBRARY#####
library(ggplot2) #help plot predicted and actual values plot, scatter plot
library(randomForest) #library for Random Forest Regression
library(corrplot) #library to create correlation plot

#####3. DATA PREPROCESSING#####
data <- read.csv("All_GPUs.csv") #Read file csv All_GPUs.csv and store into variable name "
    data"
head(data) #have the first view of All_GPUs data set

#variable name selected_data store the data set that contains only columns that we will work.
selected_data <- data[, c("Architecture", "Core_Speed", "Memory", "Memory_Bandwidth", "Memory_
    Speed", "Memory_Bus", "Memory_Type", "Process")]
head(selected_data) #have the first view of selected_data

#we can clearly see that in our data column, there are many symbols stand for NA values, so we
     store them into missingSign
missingSign <- c("","  ","\n","\n-", "\n- ")

#count the number of NA values of each column
apply(selected_data == missingSign, 2, sum)

#view the row locations of NA values of each column
apply(selected_data == missingSign, 2, which)

#compute the probability of NA value appearance of each column
apply(selected_data == missingSign, 2, mean)

#Store data into new variable name newData and then make cells contains missingSign become NA
    to easily treat in next steps
newData = selected_data
for(ch in missingSign){
  newData[newData == ch] = NA
}

#due to the low probabilities of NA values, so we decide to delete row contains NA values
    using na.omit(data_set)
newData <- na.omit(newData)

#At this step, we delete the unit in each cell, and make them become numeric type.
newData$Core_Speed <- as.numeric(sub(" MHz", "", newData$Core_Speed))

newData$Memory <- as.numeric(sub(" MB", "", newData$Memory))

newData$Memory_Bandwidth <- as.numeric(sub("GB/sec","",newData$Memory_Bandwidth))

```

```r
newData$Memory_Bus <- as.numeric(sub(" Bit", "", newData$Memory_Bus))

newData$Memory_Speed <- as.numeric(sub(" MHz", "", newData$Memory_Speed))

newData$Process <- as.numeric(sub("nm", "", newData$Process))

apply(is.na(newData), 2, sum)

summary(newData)

# Change memory_type into numeric type using factor
newData$Memory_Type <- factor(newData$Memory_Type, levels = unique(newData$Memory_Type))
newData$Memory_Type_Num <- as.numeric(newData$Memory_Type)

#Change Architecture into numeric type using factor
newData$Architecture <- factor(newData$Architecture, levels = unique(newData$Architecture))
newData$Architecture_Num <- as.numeric(newData$Architecture)

#####4. DESCRIPTIVE STATISTICS#####
#draw histogram
hist(newData$Memory_Speed, xlab = "Memory_Speed", main = "Histogram of Memory_Speed"
     , col = "orange")

hist(newData$Memory_Bandwidth, xlab = "Memory_Bandwidth", main = "Histogram of Memory_
    Bandwidth"
     , col = "orange")

hist(newData$Memory_Bus, xlab = "Memory_Bus", main = "Histogram of Memory_Bus"
     , col = "orange")

hist(newData$Memory, xlab = "Memory", main = "Histogram of Memory"
     , col = "orange")

hist(newData$Core_Speed, xlab = "Core_Speed", main = "Histogram of Core_Speed"
     , col = "orange")

hist(newData$Process, xlab = "Process", main = "Histogram of Process"
     , col = "orange")


#draw boxplot
#core_speed
boxplot(newData$Core_Speed, main = "Boxplot of Core_Speed")

#memory
boxplot(newData$Memory, main = "Boxplot of Memory")

#Memory_Bandwidth
boxplot(newData$Memory_Bandwidth, main = "Boxplot of Memory_Bandwidth")

#Memory_Speed
boxplot(newData$Memory_Speed,  main = "Boxplot of Memory_Speed")

#Memory_Bus
boxplot(newData$Memory_Bus,  main = "Boxplot of Memory_Bus")
```

```r

#Process
boxplot(newData$Process, main = "Boxplot of Process")

#Boxplot about Memory_Speed and Memory_Type_Num
boxplot(newData$Memory_Speed ~ newData$Memory_Type_Num,
        xlab = "Memory_Type_Number",
        ylab = "Memory_Speed")

#Boxplot about Memory_Speed and Process
boxplot(newData$Memory_Speed ~ newData$Process,
        xlab = "Process",
        ylab = "Memory_Speed")

#draw correlation plot
#Removed 2 character distortions
newData <- newData[, !names(newData) %in% c("Architecture", "Memory_Type")]
my_colors <- colorRampPalette(c("blue", "green", "red"))(10)  #creates a color palette consisting of 10 colors ranging from blue to red
corrplot(cor(newData), method = "number", col = my_colors, tl.cex = 0.8, number.cex = 0.8)
#The parameter method = "number" specifies that the value of each cell in the plot will be the correlation value. The col parameter is used to specify the color palette for the plot. # tl.cex and number.cex are parameters for adjusting the size of labels on the plot and numeric values, respectively.

#draw scatter plot
#Core_Speed and Memory_Speed
#shape=1 is circle
ggplot(newData, aes(x = Core_Speed, y = Memory_Speed)) +
  geom_point(shape = 1, color = "green", size = 2, alpha = 0.7) +  #shape=1 is circle
  labs( x = "Core_Speed",
        y = "Memory_Speed") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

#2: Memory and Memory_Speed
#shape=1 is circle
ggplot(newData, aes(x = Memory, y = Memory_Speed)) +
  geom_point(shape = 1, color = "green", size = 2, alpha = 0.7) +  #shape=1 is circle
  labs( x = "Memory",
        y = "Memory_Speed") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

#3: Memory_Bandwidth and Memory_Speed
#shape=1 is circle
ggplot(newData, aes(x = Memory_Bandwidth, y = Memory_Speed)) +
  geom_point(shape = 1, color = "purple", size = 2, alpha = 0.7) +
  labs( x = "Memory_Bandwidth",
        y = "Memory_Speed") +
  theme_minimal() +
```

```r
  theme(plot.title = element_text(hjust = 0.5),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

#4: Memory_Bus and Memory_Speed
#shape=1 is circle
ggplot(newData, aes(x = Memory_Bus, y = Memory_Speed)) +
  geom_point(shape = 1, color = "black", size = 2, alpha = 0.7) +
  labs( x = "Memory_Bus",
        y = "Memory_Speed") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

#5: Architecture and Memory_Speed
#shape=1 is circle
ggplot(newData, aes(x = Architecture_Num, y = Memory_Speed)) +
  geom_point(shape = 1, color = "red", size = 2, alpha = 0.7) +
  labs( x = "Architecture",
        y = "Memory_Speed") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))

#####NOTE BEFORE SWITCH TO NEXT SECTION#####
#In the next step, we store again Architecture_Num into new column name Architecture and
    delete Architecture_Num, and so to Memory_Type.

newData$Memory_Type <- newData$Memory_Type_Num
newData$Memory_Type_Num <- NULL

newData$Architecture <- newData$Architecture_Num
newData$Architecture_Num <- NULL


#####5.1. DATA PREPARATION#####
#seed(123) to make a random generation homogeneous among all our tests. Whenever we run the
    cell, we could get consistent results of the split
set.seed(123)
#This function allows us to randomly select a subset of our dataset without replacement,
    ensuring each element is chosen only once. Its output consists of the indices of 70
    randomly selected elements.
train <- sample(1:nrow(newData), nrow(newData) * 0.7)
#train_data stores the data for training
train_data <- newData[train,]
#test_data stores the data for testing
test_data <- newData[-train,]

#####5.2. LINEAR REGRESSION MODEL#####
###TRAINING###
#we first build the linear regression for Memory_Speed with all other features
lr_model <- lm(Memory_Speed ~ Architecture + Core_Speed + Memory + Memory_Bandwidth + Memory_
    Bus + Memory_Type + Process, data = train_data)
```

```r
summary(lr_model)

#Linear Regression for Memory_Speed with all other features except Memory
lr_model <- lm(Memory_Speed ~ Architecture + Core_Speed + Memory_Bandwidth + Memory_Bus + Memory_Type + Process, data = train_data)
summary(lr_model)

#draw scale location plot for linear regression model
plot(lr_model, which = 3)
#draw Q-Q plot for linear regression model
plot(lr_model, which = 2)

###TESTING###
#get predicted values and store them into predicted_value variable
predicted_value <- predict(lr_model, newdata = test_data)
#Combine predicted value and real value into a table with 2 columns Predicted and Real
predicted_real_value_table <- data.frame(Predicted = predicted_value, Real = test_data$Memory_Speed)
#print table
print(predicted_real_value_table)

#Compute accuracy of the model
accuracy <- sum(1 - abs(predicted_real_value_table$Predicted - predicted_real_value_table$Real) / predicted_real_value_table$Real) / nrow(predicted_real_value_table)
#Compute MAE of the model
mae <- sum(abs(predicted_real_value_table$Predicted - predicted_real_value_table$Real)) / nrow(predicted_real_value_table)

#print the accuracy and MAE of model
print(paste("Accuracy:", accuracy))
print(paste("MAE:", mae))

#plot predict comparing plot for linear regression model
ggplot(predicted_real_value_table, aes(x = Real, y = Predicted)) +
  geom_point(shape = 1, color = "blue") +
  labs(x = "Real Values", y = "Predicted Values") +
  geom_abline(intercept = 0, slope = 1, color = "black")


#####6.2. RANDOM FOREST MODEL#####
###TRAINING###
#rft_model store the random forest tree regression model for Memory_Speed with respect to other features except Memory
rft_model <- randomForest(Memory_Speed ~ Architecture + Core_Speed + Memory_Bandwidth + Memory_Bus + Memory_Type + Process, data = train_data)
print(rft_model)
###TESTING###
#Create a table to store predicted values and actual values. 2 columns are Predicted and Real
predicted_real_value_table_rf <- data.frame(Predicted = predict(rft_model, newdata = test_data), Real = test_data$Memory_Speed)
#print the table
print(predicted_real_value_table_rf)

#Compute accuracy, MAE and R-square of the model
```

```
accuracy <- sum(1 - abs(predicted_real_value_table_rf$Predicted - predicted_real_value_table_rf$Real) / predicted_real_value_table_rf$Real) / nrow(predicted_real_value_table_rf)
mae <- sum(abs(predicted_real_value_table_rf$Predicted - predicted_real_value_table_rf$Real)) / nrow(predicted_real_value_table_rf)
r2 <- cor(predicted_real_value_table_rf$Real, predicted_real_value_table_rf$Predicted)^2

#Print accuracy, MAE and R-Square
print(paste("Accuracy:", accuracy))
print(paste("MAE:", mae))
print(paste("R-squared:", r2))

#Plot the Q-Q plot for Random Forest model
residuals <- predicted_real_value_table_rf$Real - predicted_real_value_table_rf$Predicted
qqnorm(residuals)
qqline(residuals)

#plot predict comparing plot for random forest regression model
ggplot(predicted_real_value_table_rf, aes(x = Real, y = Predicted)) +
  geom_point(shape = 1, color = "blue") +
  labs(x = "Real Values", y = "Predicted Values") +
  geom_abline(intercept = 0, slope = 1, color = "black")

#Plot the numbers of trees by mean squared error
plot(rft_model)


#plot predict values versus actual values of 2 models
ggplot() +
  geom_point(data = predicted_real_value_table_rf, aes(x = Real, y = Predicted, color = "Random Forest"), shape = 1) +
  geom_point(data = predicted_real_value_table, aes(x = Real, y = Predicted, color = "Linear Regression"), shape = 1) +
  labs(x = "Real Values", y = "Predicted Values") + geom_abline(intercept = 0, slope = 1, color = "black")
```

References

[1] What Is a GPU? https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html

[2] Random Forest Regression – How it Helps in Predictive Analytics? https://www.analytixlabs.co.in/blog/random-forest-regression/

[3] Mean Squared Error (MSE) https://statisticsbyjim.com/regression/mean-squared-error-mse/

[4] GPU MEMORY CLOCK SPEED VS GPU CORE CLOCK SPEED, https://vibox.co.uk/blog/gpu-memory-clock-speed-vs-gpu-core-clock-speed

[5] Graphics Processing Unit (GPU): Everything You Need to Know, https://www.weka.io/learn/gpu/what-is-a-gpu/

[6] What Is Memory Clock Speed On Graphics Card, https://softwareg.com.au/blogs/computer-hardware/what-is-memory-clock-speed-on-graphics-card

[7] Multiple Linear Regression - An overview, https://www.sciencedirect.com/topics/social-sciences/multiple-linear-regression

[8] What is Random Forest?, https://ibm.com/topics/random-forest

[9] Correlogram, https://r-graph-gallery.com/correlogram.html

[10] Quantile-Quantile Plot , https://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.html

[11] What Is Analysis of Variance (ANOVA)? , https://www.investopedia.com/terms/a/anova.asp

[12] A comparison of multiple linear regression and random forest for community concern of youth and young adults survey , https://sjst.psu.ac.th/journal/44-2/26.pdf

[13] Parameter estimates, https://analyse-it.com/docs/user-guide/fit-model/linear/parameter-estimates